# Crowdsourcing security

## Lessons in open code and bug bounties

Colin Percival
cperciva@tarsnap.com

May 18, 2012

# Who am I?

- FreeBSD Security Officer since 2005.
  - Responsible for handling $\approx 140$ security advisories in the FreeBSD base system.
  - FreeBSD is a free UNIX-like operating system based on BSD UNIX (4.4BSD-Lite2).
    - We're not allowed to say that FreeBSD is UNIX because we haven't paid the trademark owner.
  - Project is 19 years old, has 200+ active source code committers, 9.3 MLOC.
  - Volunteer position.
- Founder of the Tarsnap online backup service.
  - Started in 2006, one-man company, 73 kLOC.
  - As FreeBSD Security Officer, I needed my backups to be secure, and I didn't trust any existing options.
  - My day job.

# Tarsnap

- Tarsnap is not open source software . . .
  - Commercial reality: Most of the intelligence in Tarsnap is client-side, and I don't want to compete against my own code.
  - Tarsnap is built using the "libarchive" (BSD licensed) — the BSD license permits closed-source derivative works.
  - Tarsnap contributes bug fixes and non-core features back to libarchive and spins off other code.
- . . . but the client application source code is available for users to inspect and compile themselves anyway.
  - Tarsnap is "Online backups for the truly paranoid".
  - If you're truly paranoid, you don't trust opaque binaries.

# Oops

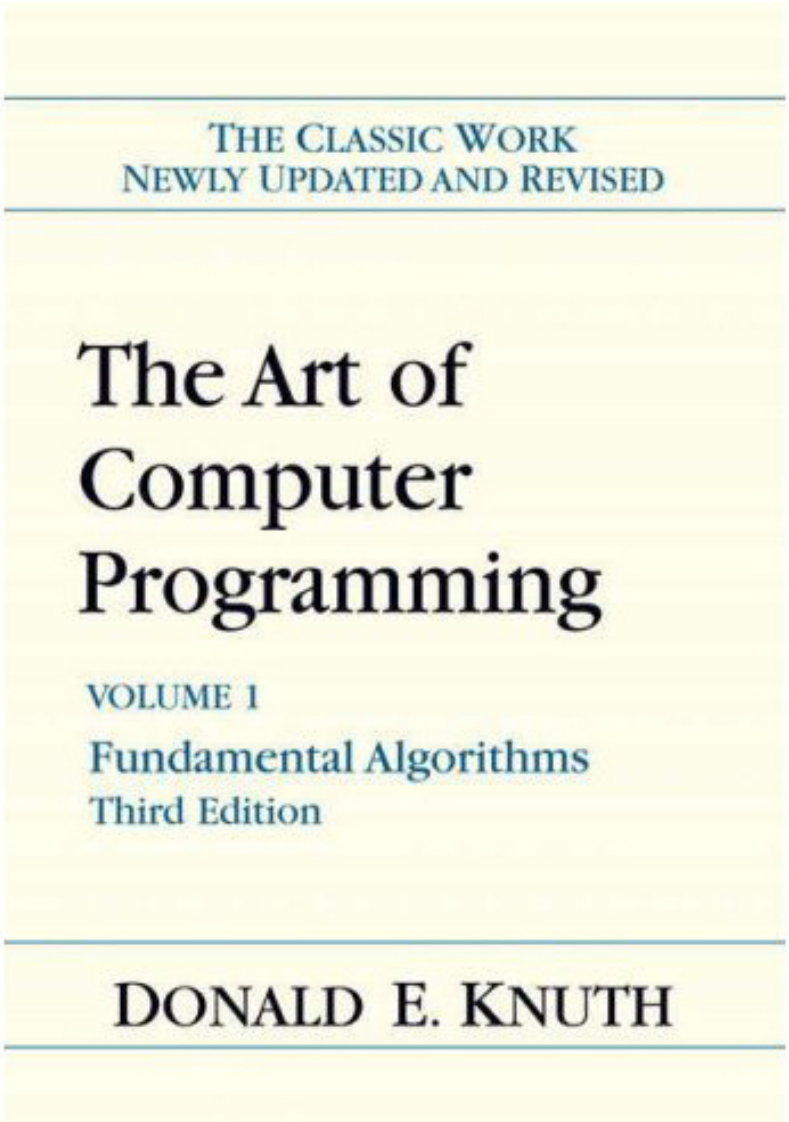
- In January 2011, I received an email: "I was a little confused by a part of the crypto . . .".
  - Serious cryptographic bug: Tarsnap was reusing encryption nonces.
  - Under certain conditions I might be able to read someone's archived data.
  - It turned out that my original code from June 2007 was correct, but in April 2009 I lost a ++ when I refactored the code.
- The bug was found by someone who was reading the code *out of curiosity*.

# Serendipity

- "The most exciting phrase to hear in science, the one that heralds new discoveries, is not 'Eureka!' but 'That's funny. . .'."
- Most FreeBSD security vulnerabilities were not found by people who were looking for them.
  - At least half were "I was looking at this code and I noticed that this looked wrong".
  - Many more were "I was tracking down a bug, and when I found it I saw that it could be a security vulnerability".
  - Out of over 140 FreeBSD security advisories, I only know of 2 which were exploited in the wild before our advisory went out.
- I found the 2005 Intel HyperThreading vulnerability because I was reading an optimization manual.

# Bug bounties

- In order to get more people looking at Tarsnap code, I decided to offer bug bounties.
  - Traditionally bug bounty programs have only offered prizes for security vulnerabilities.
  - Problem: "I think any reviewer who wanted to get paid would not start with Colin's code as an easy place to find bugs."
  - Solution: Make it easier for people to win bounties.
- I decided to offer bounties for all errors in my code.
  - Up to $2000 for a new security bug.
  - Down to $1 for a typographical error in a source code comment.


THE CLASSIC WORK
NEWLY UPDATED AND REVISED
The Art of Computer Programming
VOLUME 1
Fundamental Algorithms
Third Edition
DONALD E. KNUTH

# Software *Engineering*

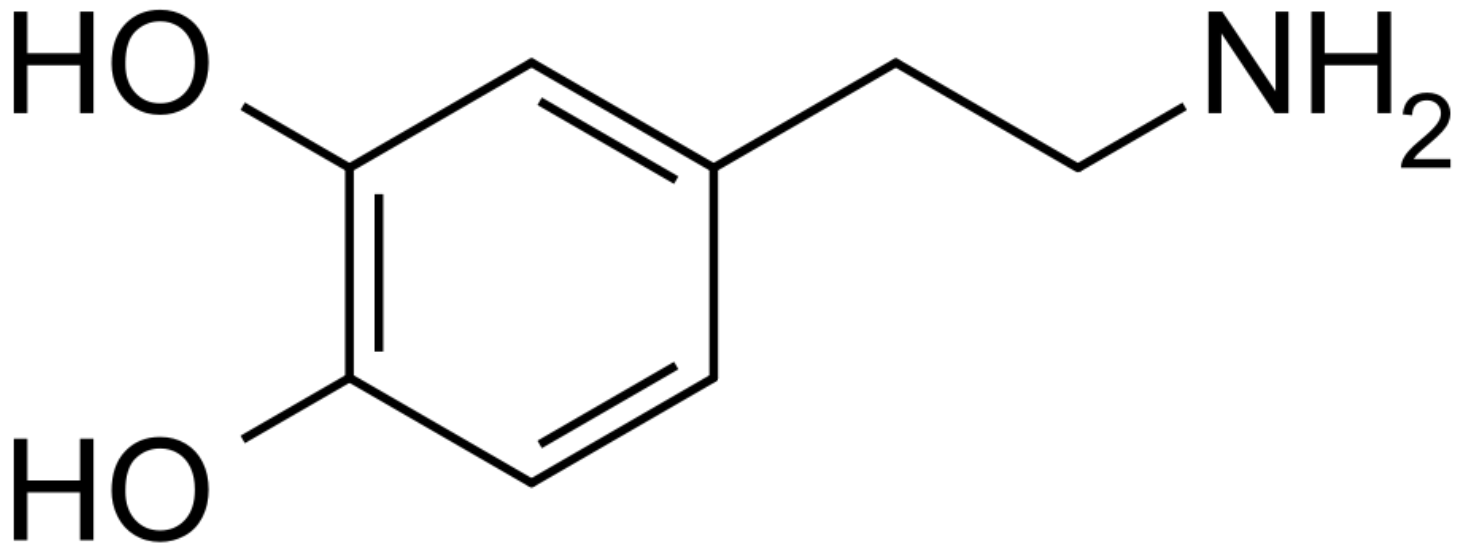
- If software is an engineering field, we should pay attention to lessons learned from other engineering fields.
- Industrial safety engineering has the concept of an *accident pyramid*.
- Observation by H. W. Heinrich in 1931: For every serious injury, there are . . .
  - . . . 10 minor injuries.
  - . . . 30 incidents causing property damage.
  - . . . 600 near misses.
  - . . . 6000 unsafe behaviours.
- In order to prevent serious injuries, *target the unsafe behaviours*.

# Tarsnap bug pyramid

- Only 1 major security vulnerability, but . . .
  - . . . 3 minor security vulnerabilities.
  - . . . 12 user-visible misbehaviours.
  - . . . 71 instances of harmlessly-wrong code.
  - . . . 155 cosmetic errors in code.
- Most bugs could have been worse if the surrounding code was different.
  - Memory leaks... in error-handling paths which result in `exit(1)` a few microseconds later.
  - Website vulnerable to cross-site scripting... but only by a logged-in user, against himself.
  - Library code has a buffer overflow... but only on input values which never get passed to it.
- If I didn't fix these minor bugs, they could *become* security vulnerabilities at a later time.

# FreeBSD security non-vulnerabilities

- FreeBSD has had a lot of "lucky" non-vulnerabilities too.
  - Vulnerabilities evaded by implementation details: "That's a buffer overflow... into memory which is never used due to memory alignment requirements."
  - Vulnerabilities in dead code: "That's a bug... in a function which is never used."
  - Bugs eating bugs: "This is a remote privilege escalation bug... in a feature which was accidentally broken ten years ago and now crashes if you try to use it."
- All of these could easily become security vulnerabilities at a later time if not fixed first.
- Accident pyramid: To reduce workplace injuries, target *unsafe behaviours*.
- Software bug pyramid: To reduce security vulnerabilities, target *bad code*.




Dopamine

# Dopamine

- Dopamine is released in the brain in response to unexpected rewards.
  - Dopamine plays a role in mediating addictive behaviours.
  - Parkinson's patients treated with dopamine agonists often become addicted to gambling.
  - If you give rats access to dopamine, they will behave very irrationally.
- Looking for bugs has a similar reward profile to gambling.
  - Frequent $1 bug bounties mixed with occasional $10 and $50 bug bounties.
  - After a while, the mental addiction supplements the cash value of the bug bounties.
- Highly skilled developers will work for $10 / hour if you tell them that they're winning prizes!

# Casual code review

- Crowdsourced code review is *casual* code review.
  - People look at what they find interesting.
  - You can't fire people for not reviewing the code you think needs to be reviewed.
  - Worse than writing open source software: You don't even know which code has been inspected.
- Did you know that telnet has support for encryption?
  - FreeBSD-SA-11:08.telnetd, December 23, 2011: Buffer overflow in encryption code in BSD telnetd.
  - The bug was probably written as part of MIT Kerberos in 1990, and was introduced to BSD in March 1991.
  - Anyone with security experience looking at the code in the past decade would have noticed the buffer overflow...
  - ... but nobody ever did, because we all use SSH now.

# Casual code review

- Casual code readers don't look at ugly code.
  - They're usually optimizing for happiness, and ugly code makes people sad.
  - If you want more people to read your code, *make your code readable*.
- Experiment: Divide FreeBSD source code into 50% "stylish" files and 50% "non-stylish" files based on consistency with `indent(1)`.
  - Stylish and non-stylish files are equally likely to be involved in a security advisory.
  - ... but security bugs in non-stylish files are present on average 4 $\times$ longer before they are found and fixed.
  - Ugly code has more bugs but gets less attention!

# Casual code review

- Casual code readers seek instant gratification.
  - Often people will be reading your code in quanta of 10 minutes or less.
  - If a block of code is large or complicated, they will move on to a simpler piece of code.
- ~~"Always code as if the person who ends up maintaining your code is a violent psychopath who knows where you live." — John F. Woods~~
- Always code as if the person who will end up reviewing your code is an intern with ADHD who forgot to take his Ritalin.
- Excessively explicit comments can help here.

```
/* Add two to i.  */
i++;
```

# Casual code review

- Casual code readers (probably) aren't domain experts.
  - Simple statistics: Most people aren't domain experts.
  - You shouldn't expect to receive very much useful *design* review from the crowd.
  - You'll probably get lots of design review, but most of it will be hopelessly inaccurate.
- Not really a big problem, since you should be able to review your design sufficiently in-house.
  - You should have relevant domain expertise already.
  - The design should be shorter and less time-consuming to review than the code which implements it.
  - If you're designing a cryptographic protocol specified by a 104 page long RFC, you're doing it wrong.

# Conclusions

- It may be worth publishing source code even if you can't or don't want to release it under an open source license.
- If your company publishes source code, you should offer bug bounties.
  - Don't think of this as an added cost; think of it as a source of cheap developer hours.
- If you want to produce secure code, sweat the small stuff.
- If you want to benefit from crowd-sourced code review, writing good, clean, well-designed code is even more important than normal.

# Questions?